REMARKS

This Amendment is filed in response to the FINAL amendment mailed on June 25, 2007, and in the Request for Continued Examination (RCE) filed on even date here-with. All objections and rejections are respectfully traversed.

Claims 1-37 are in the case.

Claims 1, 11, 17, 25, 28, and 34 were amended.

No claims were added.

Request for Interview

The Applicant respectfully requests a telephonic interview with the Examiner after the Examiner has had an opportunity to consider this Amendment, but before the issuance of the next Office Action. The Applicant may be reached at 617-951-3028.

At Paragraph 3 of the Office Action Claims 1-37 were rejected under 35 U.S.C. 103(a) as being unpatentable over Gillespie U. S. Patent No. 6,269,391 issued July 31, 2001.

Applicant's claimed invention, as set forth in representative claim 1, comprises in part:

1. A method for executing uniprocessor (UP) coded workloads in a multiprocessor (MP) computer system without having to rewrite the UP-coded work-loads' code, comprising:

identifying threads in the uniprocessor coded workloads (UP-workloads) which can execute concurrently, and identifying threads in the UP-workloads which cannot execute concurrently;

assigning first threads which cannot execute concurrently to a first concurrency group, and assigning second threads which cannot execute concurrently to a second concurrency group, where any thread in the first concurrency group can execute concurrently with any thread in the second concurrency group;

compiling the UP workload to execute on the MP system, the threads in the first concurrency group compiled to be prevented from executing concurrently, the threads in the second concurrency group compiled to be prevented from executing concurrently, and the threads in the first and the second concurrency groups compiled to execute concurrently;

scheduling first and second execution vehicles that respectively execute on different processors in the MP computer system at substantially the same time;

acquiring the first concurrency group by the first execution vehicle and the second concurrency group by the second execution vehicle; and

executing UP-coded workloads in the first concurrency group through the first execution vehicle at substantially the same time as UP-coded workloads in the second concurrency group are executed through the second execution vehicle.

Gillespie discloses a scheduling kernel for scheduling several virtual machines on a multiprocessor. Gillespie organizes threads to be processed into an execution exclusion set, where the execution exclusion set is a group of threads or virtual machines, none of which may be processed in parallel. (Col. 6, lines 50-53). Additionally, the execution exclusion set are monitored by a execution exclusion set module to limit execution to a single thread at a time out of any particular execution exclusion set of threads.

Applicant respectfully urges that Gillespie has no disclosure of Applicant's claimed novel ***identifying threads in the uniprocessor coded workloads (UP-workloads) which can execute concurrently, and identifying threads in the UP-workloads which cannot execute concurrently;***

assigning first threads which cannot execute concurrently to a first concurrency group, and assigning second threads which cannot execute concurrently to a second concurrency group, where any thread in the first concurrency group can execute concurrently with any thread in the second concurrency group;

compiling the UP workload to execute on the MP system, the threads in the first concurrency group compiled to be prevented from executing concurrently, the threads in the second concurrency group compiled to be prevented from executing concurrently, and the threads in the first and the second concurrency groups compiled to execute concurrently.

Applicant respectfully urges that Gillespie has no disclosure of Applicant's claimed ***identifying threads in the uniprocessor coded workloads (UP-workloads) which can execute concurrently, and identifying threads in the UP-workloads which cannot execute concurrently***. That is, Gillespie has no disclosure of Applicant's claimed ***identifying threads in the uniprocessor coded workloads***, and further Gillespie has no disclosure of ***uniprocessor coded workloads***.

Further, Applicant respectfully urges that Gillespie has no disclosure of Applicant's claimed ***compiling the UP workload to execute on the MP system, the threads in the first concurrency group compiled to be prevented from executing concurrently, the threads in the second concurrency group compiled to be prevented from executing concurrently, and the threads in the first and the second concurrency groups compiled to execute concurrently***. That is, Gillespie has no disclosure of Applicant's ***compiling the UP workload to execute on the MP system***.

Even further, Applicant claims ***identifying threads in the uniprocessor coded workloads*** and then compiling the threads identified so that ***the threads in the first concurrency group compiled to be prevented from executing concurrently, the threads in the second concurrency group compiled to be prevented from executing concurrently, and the threads in the first and the second concurrency groups compiled to execute concurrently***.

Applicant respectfully urges that Gillespie has no disclosure of converting uniprocessor workloads into multiprocessor workloads, as claimed in representative claim 1.

Accordingly, Applicant respectfully urges that Gillespie is legally incapable of rendering Applicant's claimed invention unpatentable under 35 U.S.C. 103(a) because of the absence from Gillespie's disclosure of Applicant's claimed novel

identifying threads in the uniprocessor coded workloads (UP-workloads) which can execute concurrently, and identifying threads in the UP-workloads which cannot execute concurrently;

assigning first threads which cannot execute concurrently to a first concurrency group, and assigning second threads which cannot execute concurrently to a second concurrency group, where any thread in the first concurrency group can execute concurrently with any thread in the second concurrency group;

compiling the UP workload to execute on the MP system, the threads in the first concurrency group compiled to be prevented from executing concurrently, the threads in the second concurrency group compiled to be prevented from executing concurrently, and the threads in the first and the second concurrency groups compiled to execute concurrently.

All independent claims are believed to be in condition for allowance.

All dependent claims are believed to be dependent from allowable independent claims. Accordingly, all claims are believed to be in condition for allowance.

Please charge any additional fee occasioned by this paper to our Deposit Account No. 03-1237.

Respectfully submitted,

/A. Sidney Johnston/

A. Sidney Johnston
Reg. No. 29,548
CESARI AND MCKENNA, LLP
88 Black Falcon Avenue
Boston, MA 02210-2414
(617) 951-2500